# Programming Guide for EETI MultiTouch ( EXC7200 ) I2C interface

## Software Protocol

I2C Transaction Frame: each I2C transaction frame transfers one I2C packet data. Each I2C packet data may not be an exact application packet.

From Host to Device:

Report ID = 3 ( Diagnostics mode )

| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Byte9 |
|---|---|---|---|---|---|---|---|---|---|
| 0x03 | len | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |

len = valid data length in bytes of this current I2C data packet.

D1 to DN totally N bytes are valid data in this current I2C data packet.

N <= 8. This Report ID = 3 is used for diagnostics

From Device To Host:

Host computer poll this I2C device only when the IRQ pulled low by this Touch device. The host computer should not poll this device when this IRQ signal pulled high.

The packet size of each I2C transaction frame is defined as 12 bytes

| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Byte9 |
|---|---|---|---|---|---|---|---|---|---|
| ID | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |

ID is defined as Report ID. The report ID was defined as below

Report ID = 1 ( single touch mouse mode )

| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Byte9 |
|---|---|---|---|---|---|---|---|---|---|
| 0x01 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |

D0 = Mouse Button States

D1 = Low byte of X coordination

D2 = High byte of X coordination

D3 = Low byte of Y coordination

D4 = High byte of Y coordination

[D5..D8] not used and must be kept as 0

Report ID = 3 ( Diagnostics mode )

| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Byte9 |
|---|---|---|---|---|---|---|---|---|---|
| 0x03 | len | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |

len = valid data length in bytes of this current I2C data packet.

D1 to DN totally N bytes are valid data in this current I2C data packet.

N <= 8.

This Report ID is used for diagnostics

Report ID = 4 ( MutliTouch report )

| Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Byte8 | Byte9 |
|---|---|---|---|---|---|---|---|---|---|
| 0x04 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |

D0 : B7 = Touch Valid. B7 = 1 is valid touch

[B6:B2] = Contact ID.

B1 = In Range bit, this bit should be always 1

B0 = Down/Up bit, B0 = 1 for Touch Down, B0 = 0 for Lift Off

D1 = Low byte of X coordination

D2 = High byte of X coordination

D3 = Low byte of Y coordination

D4 = High byte of Y coordination

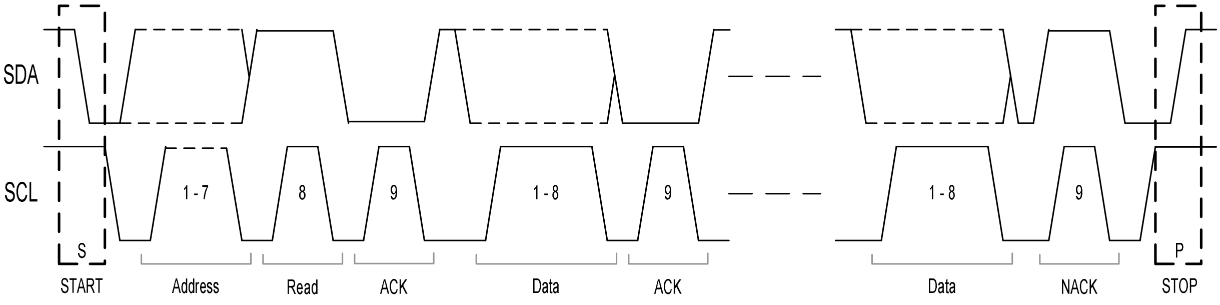
D5 = Low byte of Z coordination

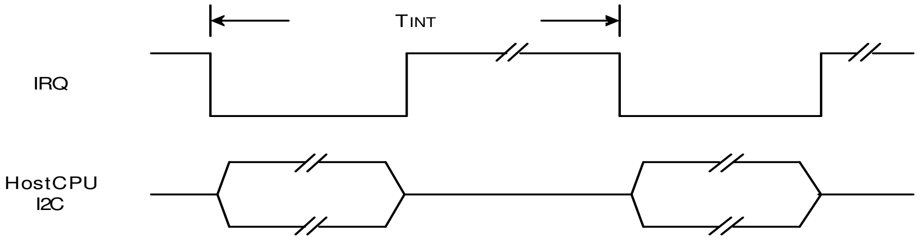
D6 = High byte of Z coordination

D7,D8 = reserved and must be zero.

**Hardware Interface**

I2C Timing Chart:





## I/O Pin Definitions

| Pin Number | Pin Name | I/O | Description |
|---|---|---|---|
| 1 | GND | Power | Ground |
| 2 | SDA | Open Drain | I2C Data |
| 3 | SCL | Open Drain | I2C Clock |
| 4 | VDD | Power | 5V/3.3V regulated |
| 5 | IRQ | Open Drain | Interrupt Request pin |
| 6 | RST | Input/Open Drain | Reset pin to Master Chip |

SDA:

I2C data pin. This pin is configured as open drain. It needs a pull up resistor( 4.7K ) for I2C communication.

SCL:

I2C Clock pin. This pin is configured as open drain pin. It needs a pull up resistor( 4.7K) for I2C communication.

VDD:

The power supply pin.

GND:

Ground pin. This pin should be connected with the host GND

IRQ:

Interrupt request pin. This pin should be kept at logic high at idle state. Whenever the controller has any data to be sent to host computer, controller pulls low it to generate an interrupt to host computer.
When in Idle state or Sleep state, host computer can wake up the controller via an falling edge on IRQ pin to wake up the device.

RST:

Optional reset pin. Pull low this pin and up this pin to cause Touch controller reset.

**Power Saving Mechanism**

EXC72XX-I supports 3 working mode for power saving.

Fully Working:

After reset, the controller module works at full power working state. It needs around 50mA at this mode.

Idle:

After EXC72XX-II receives a software packet from host computer to request MCU entering Idle state. This controller module sends back an Idle ready packet data to host computer when it is ready to Idle.

At Idle state, IRQ pin will be released to high state. Host computer can wake up this controller modules via generating an falling edge signal @ IRQ pin.

During Idle state, controller scan touch sensor at low speed – around once 100 ms and can be controlled via host computer. Once it detects sensor touched, the controller will back to fully working state automatically.

Working at this mode, the power consumption should be around 2mA.

Sleep:

Whenever the host computer wants to deep sleep, it issue a Sleep command packet to this controller. Once the controller firmware receives such Sleep command, it enters deep sleep state. Only host computer can wake up this device via generating a falling edge signal at IRQ pin. Working at Sleep state, the power consumption should be around or less than 300uA.

**Commands**

1**. Set Idle State**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| H->D | 0x03 | 0x06 | 0x0A | 0x04 | 0x36 | 0x3F | 0x01 | T | 0 | 0 |
| D->H | 0x03 | 0x06 | 0x0A | 0x04 | 0x36 | 0x3F | 0x01 | t | 0 | 0 |

Host computer send the command and touch device response as above for Idle state configuration setting.

Where, T means the scanning interval when in idle state. The touch controller will wakeup every that period of time to scan touch screen to check if the touchscreen touched or not. Once it detects touchscreen touched, it will back to fully working state immediately. The reasonable value is 0~9. The interval = ( T +1) X 50ms.   t   in the response packet means the most recent scanning time interval setting value.

### 2. Sleep State

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| H->D | 0x03 | 0x06 | 0x0A | 0x03 | 0x36 | 0x3F | 0x02 | 0 | 0 | 0 |
| D->H | 0x03 | 0x06 | 0x0A | 0x03 | 0x36 | 0x3F | 0x02 | 0 | 0 | 0 |

Host computer send above command packet to touch controller device to make the device enter sleep state for power saving. Once the touch controller device receives this command packet, it enters deep sleep state and the controller does not response until it wakes up from this sleep state. The touch controller device response with above D->H packet data only after it wakes up.

### Command to Query firmware version

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| H->D | 0x03 | 0x03 | 0x0A | 0x01 | ‘D’ | 0 | 0 | 0 | 0 | 0 |
| D->H | 0x03 | 0x06 | 0x0A | 0x06 | ‘D’ | ‘1’ | ‘.’ | ‘0’ | ‘0’ | 0x00 |

Host computer send above H->D command packet to touch controller device to query the firmware version. The device responses with the data packet like above D->H packet. However, the device responses to this query command with the firmware version ASCII string. The length of the firmware version is variable. For above example, the version string is “1.00”.